Preprocessing

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


%matplotlib inline
```

In [2]:
```
liver_df = pd.read_csv('liverLabTrain.csv')
```

In [3]:
```
liver_df.head(10)
```

Out[3]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase
0	65	Female	0.7	0.1	187	16
1	62	Male	10.9	5.5	699	64
2	62	Male	7.3	4.1	490	60
3	58	Male	1.0	0.4	182	14
4	72	Male	3.9	2.0	195	27
5	46	Male	1.8	0.7	208	19
6	26	Female	0.9	0.2	154	16
7	29	Female	0.9	0.3	202	14
8	51	Male	2.9	1.3	482	22
9	62	Male	6.8	3.0	542	116

In [4]:
```
from sklearn.preprocessing import OneHotEncoder
```

Binarizing the gender column

In [5]:
```
for i,x in enumerate(liver_df['Gender']):

        if x.lower() == 'female':

            liver_df['Gender'].iloc[i] = 0

        else:

            liver_df['Gender'].iloc[i] = 1
```

```
c:\program files\python37\lib\site-packages\pandas\core\indexing.py:670: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  self._setitem_with_indexer(indexer, value)
```

In [6]:
```
liver_df.head(5)
```

Out[6]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase
0	65	0	0.7	0.1	187	16
1	62	1	10.9	5.5	699	64
2	62	1	7.3	4.1	490	60
3	58	1	1.0	0.4	182	14
4	72	1	3.9	2.0	195	27

In [7]:
```
liver_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 483 entries, 0 to 482
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Age                         483 non-null    int64  
 1   Gender                      483 non-null    object 
 2   Total_Bilirubin             483 non-null    float64
 3   Direct_Bilirubin            483 non-null    float64
 4   Alkaline_Phosphotase        483 non-null    int64  
 5   Alamine_Aminotransferase    483 non-null    int64  
 6   Aspartate_Aminotransferase  483 non-null    int64  
 7   Total_Protiens              483 non-null    float64
 8   Albumin                     483 non-null    float64
 9   Albumin_and_Globulin_Ratio  480 non-null    float64
 10  Liver_Disease               483 non-null    int64  
dtypes: float64(5), int64(5), object(1)
memory usage: 41.6+ KB
```

In [8]:
```
liver_df.describe()
```

Out[8]:

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase
count	483.000000	483.000000	483.000000	483.000000	483.000000
mean	44.722567	3.299172	1.466253	287.335404	72.111801
std	16.263700	6.358002	2.783368	232.322630	148.754051
min	4.000000	0.400000	0.100000	75.000000	10.000000
25%	33.000000	0.800000	0.200000	174.500000	23.000000
50%	45.000000	1.000000	0.300000	206.000000	34.000000
75%	57.000000	2.600000	1.250000	298.000000	58.000000
max	90.000000	75.000000	19.700000	2110.000000	1680.000000

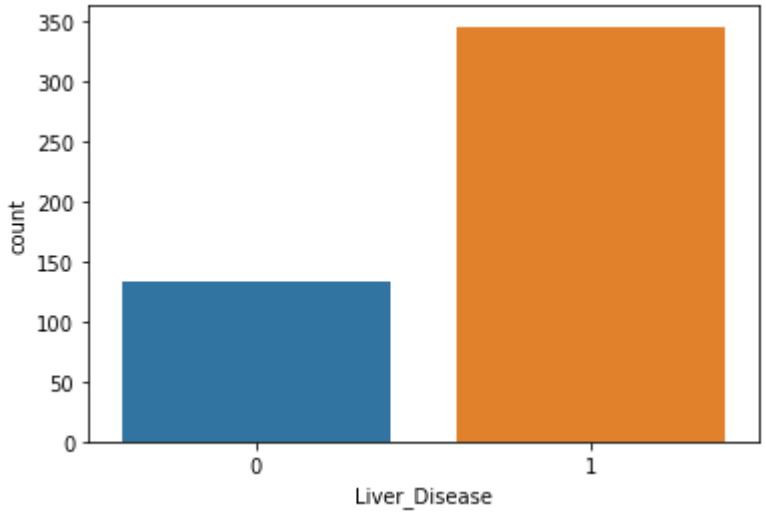
Missing data

In [9]:
```
liver_df.dropna(inplace=True) #Dropping 3 rows where Albumin_and_Globulin_Ratio i
```

Visualizations

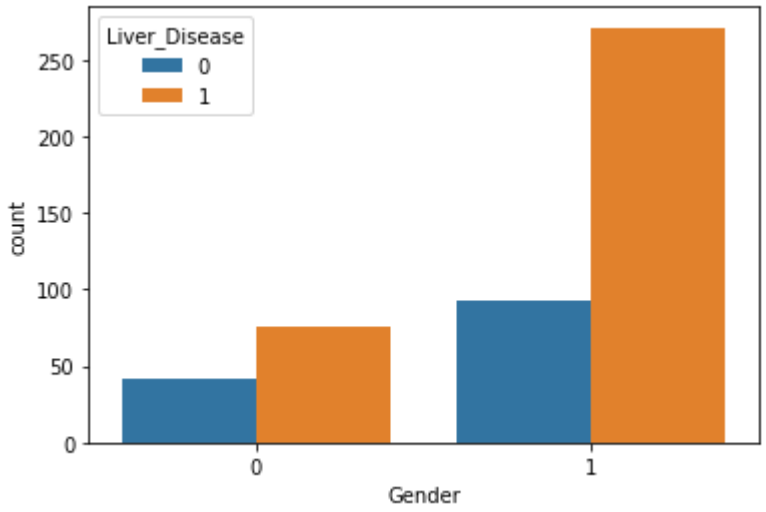
In [10]:
```
sns.countplot(liver_df['Liver_Disease'])
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b796aa4148>
```



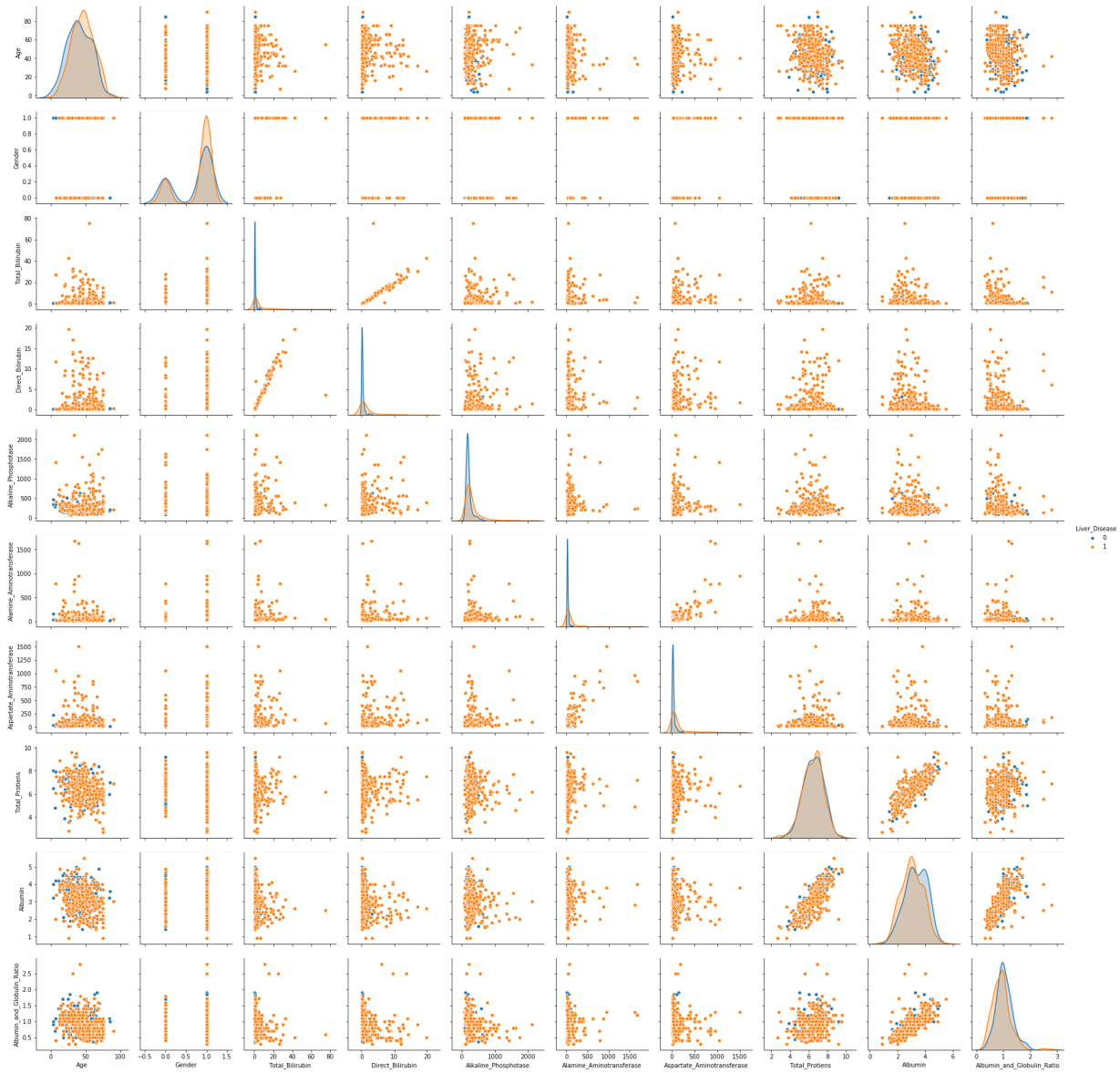
In [11]:
```
sns.countplot(data = liver_df, x = 'Gender', hue='Liver_Disease')
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b798b93f48>
```



In [13]:
```
sns.pairplot(data = liver_df, hue = 'Liver_Disease')
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x2b7a090af08>
```



Preparing data for model

In [14]:
```
X = liver_df.drop(['Liver_Disease'], axis = 1)
```

In [15]:
```
y = liver_df['Liver_Disease']
```

In [16]:
```
X
```

Out[16]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase
0	65	0	0.7	0.1	187	16
1	62	1	10.9	5.5	699	64
2	62	1	7.3	4.1	490	60
3	58	1	1.0	0.4	182	14
4	72	1	3.9	2.0	195	27
...	...	...	...	...	...	..
478	60	1	0.5	0.1	500	20
479	40	1	0.6	0.1	98	35
480	52	1	0.8	0.2	245	48
481	31	1	1.3	0.5	184	29
482	38	1	1.0	0.3	216	21

480 rows × 10 columns

In [17]:
```
y
```

Out[17]:
```
0      1
1      1
2      1
3      1
4      1
      ..
478    0
479    1
480    1
481    1
482    0
Name: Liver_Disease, Length: 480, dtype: int64
```

In [18]:
```
from sklearn.model_selection import train_test_split
```

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
```

Oversampling

In [37]:
```
from imblearn.over_sampling import SMOTE
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-37-0b052d41b57a> in <module>
----> 1 from imblearn.over_sampling import SMOTE

~\AppData\Roaming\Python\Python37\site-packages\imblearn\__init__.py in <module
>
     32     Module which allowing to create pipeline with scikit-learn estimato
rs.
     33 """
---> 34 from . import combine
     35 from . import ensemble
     36 from . import exceptions

~\AppData\Roaming\Python\Python37\site-packages\imblearn\combine\__init__.py in
<module>
      3 """
      4
----> 5 from ._smote_enn import SMOTEENN
      6 from ._smote_tomek import SMOTETomek
      7

~\AppData\Roaming\Python\Python37\site-packages\imblearn\combine\_smote_enn.py
 in <module>
      8 from sklearn.utils import check_X_y
      9
---> 10 from ..base import BaseSampler
     11 from ..over_sampling import SMOTE
     12 from ..over_sampling.base import BaseOverSampler

~\AppData\Roaming\Python\Python37\site-packages\imblearn\base.py in <module>
     14 from sklearn.utils.multiclass import check_classification_targets
     15
---> 16 from .utils import check_sampling_strategy, check_target_type
     17 from .utils._validation import ArraysTransformer
     18

~\AppData\Roaming\Python\Python37\site-packages\imblearn\utils\__init__.py in <
module>
      5 from ._docstring import Substitution
      6
----> 7 from ._validation import check_neighbors_object
      8 from ._validation import check_target_type
      9 from ._validation import check_sampling_strategy

~\AppData\Roaming\Python\Python37\site-packages\imblearn\utils\_validation.py i
n <module>
     11
     12 from sklearn.base import clone
---> 13 from sklearn.neighbors._base import KNeighborsMixin
     14 from sklearn.neighbors import NearestNeighbors
     15 from sklearn.utils import column_or_1d

c:\program files\python37\lib\site-packages\sklearn\neighbors\_base.py in <modu
le>
```

```
     25 from ..metrics.pairwise import PAIRWISE_DISTANCE_FUNCTIONS
     26 from ..utils import check_X_y, check_array, gen_even_slices
---> 27 from ..utils import _to_object_array
     28 from ..utils.multiclass import check_classification_targets
     29 from ..utils.validation import check_is_fitted

ImportError: cannot import name '_to_object_array' from 'sklearn.utils' (c:\pro
gram files\python37\lib\site-packages\sklearn\utils\__init__.py)
```

Using GridSearchCV to find the best parameters for Logistic Regression

In [22]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
```

In [21]:
```
estimators = [20, 40, 60, 80, 100]
criterion = ['gini', 'entropy']
min_samples_split = [2,3,4]
max_features = ['auto', 'sqrt']



# Create hyperparameter options
hyperparameters = dict(n_estimators=estimators,criterion = criterion,min_samples_
```

In [23]:
```
rf = RandomForestClassifier(verbose=1, random_state=1)
```

In [24]:
```
clf = GridSearchCV(rf, hyperparameters, cv=5, verbose=0)
```

In [25]:
```
best_model = clf.fit(X_train, y_train)
```

```
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
```

In [26]:
```
print('Best n_estimators:', best_model.best_estimator_.get_params()['n_estimators
print('Best criterion:', best_model.best_estimator_.get_params()['criterion'])
print('Best max_features:', best_model.best_estimator_.get_params()['max_features
print('Best min_samples_split:', best_model.best_estimator_.get_params()['min_sam
```

```
Best n_estimators: 40
Best criterion: entropy
Best max_features: auto
Best min_samples_split: 3
```

Training with the best parameters

In [29]:
```
tuned_rf = RandomForestClassifier(n_estimators=40,
                                  criterion='entropy',
                                  max_features='auto'
                                  ,min_samples_split=3)
```

In [30]:
```
tuned_rf.fit(X_train,y_train)
```

Out[30]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=Non
e,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=3,
                       min_weight_fraction_leaf=0.0, n_estimators=40,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [31]:
```
y_preds = tuned_rf.predict(X_test)
```

In [32]:
```
from sklearn.metrics import classification_report
```

In [33]:
```
print(classification_report(y_test,y_preds))
```

```
              precision    recall  f1-score   support

           0       0.65      0.45      0.53        29
           1       0.79      0.90      0.84        67

    accuracy                           0.76        96
   macro avg       0.72      0.67      0.68        96
weighted avg       0.75      0.76      0.75        96
```

In [ ]:
```

```